



Multi Core SSL/TLS Security Processor Architecture Prototype Design with automated Preferential Algorithm in FPGA

Rourab Paul¹ Amlan Chakrabarti² and Ranjan Ghosh³

A.K.Choudhury School of Information Technology^{1,5}, Dept. of Computer Science and Engineering² and Institute of Radio Physics and Electronics^{3,4,6}, University of Calcutta, 92 A. P. C. Road, Kolkata-700 009, India.

Abstract

In this paper a pipelined architecture of a high speed network security processor (NSP) for SSL/TLS protocol is implemented on a system on chip (SOC) where hardware information of all encryption, hashing and key exchange algorithms are stored in flash memory in terms of bit files, in contrary to related works where all are actually implemented in hardware. The NSP finds applications in e-commerce, virtual private network (VPN) and in other fields that require data confidentiality. The motivation of the present work is to dynamically execute applications with stipulated throughput within budgeted hardware resource and power. A preferential algorithm choosing an appropriate cipher suite is proposed, which is based on Efficient System Index (ESI) budget comprising of power, throughput and resource given by the user. The bit files of the chosen security algorithms are downloaded from the flash memory to the partial region of field programmable gate array (FPGA). The proposed SOC controls data communication between an application running in a system through a PCI and the Ethernet interface of a network. Partial configuration feature is used in ISE14.4 suite with ZYNQ 7z020-clg484 FPGA platform. The performances of the implemented crypto algorithms are considerably better than the existing works reported in literatures.

© 2014 Manuscript in Elsevier Ltd format.

Keywords: Cryptography, Multicipher, Multicore, Hardware Design, FPGA, SSL.

1. Introduction

Security protocols, SSL, TLS and IPSec are executed in software providing run time flexibility, but executing in large time. For better throughput, many hardware architectures are proposed in literatures [1], [2], [3], [4] and [5], where the flexibility of crypto algorithms is achieved by making all algorithms available in FPGA-ASIC platform during its entire runtime. The combination of algorithms, selected for a particular session gets activated by an enable signal deactivating others allowing the deactivated ones to continue consuming resources and power. Reducing the power-resource metrics and simultaneously preserving the algorithmic flexibility is a serious challenge in embedded systems. In this paper, keeping the challenging issues in mind, it is proposed to store bit files of all necessary algorithms (encryption, hashing and key exchange algorithms) in the flash memory, instead of placing them in the reconfigurable cells (distributed RAM) of FPGA [6]. The bit files of the crypto suite chosen following a preferential algorithm, proposed in the paper, are only configured in the reconfigurable cells of the partitioned partial region of the FPGA saving a huge hardware slices and system power consumption. The contributions in this paper are as follows: (I) proposition of an NSP architecture using two ARM cortex processors, two crypto engines and 2 DMAs, (II) communication between system and network using two interfaces, PCI and Ethernet, (III) data from PCI and Ethernet are processed in parallel by two sets of processing elements, DMAs and crypto engines, (IV) 7 encryption algorithms,

3 hash algorithms and 3 key exchange algorithms are designed and stored in the flash memory as bit files after undertaking a thorough studies of the issues of power, resource and throughput of these algorithms, (V) a preferential algorithm is proposed to choose appropriate encryption, hash and key exchange algorithms according to the budgeted ESI comprising of power, throughput and resource or any combination of them.

The organization of this paper is as follows. In Section II we have described preferential algorithm, Section III details out the proposed system overview. Section IV and V deals with the architectural topology of the proposed design and result-implementations respectively, and finally we conclude in Section VI.

2. Multi Metric Preferential Algorithm

Multi metric algorithm is a technique in proposed SSL/TLS architecture, which can suggest one or more than one cipher suite combinations according to a budgeted system metrics of the user. If any cipher suite combination can not meet the budget, system will ask the user to increase metrics budget. According to the application requirements of crypto embedded systems 5 modes of priority should be there.

- *Power Priority Mode*: In battery powered wireless devices it is needed to maintain the power budget at its highest priority over rest of the metrics namely throughput and resource. This mode will help to choose those cipher suites, which meets the user power budget without aware of rest of the metrics.
- *Throughput Priority Mode*: In this mode throughput budget is at highest priority over rest of the metrics i.e. power and resource. It finds application in high speed data communication system.
- *Resource Priority Mode*: In this mode the resource budget is at highest priority over rest of the metrics i.e. power and throughput. For low cost embedded systems like RFID scanner it is required to reduce the resource usage as much as possible.
- *Priority Mode*: This mode does not have a single metric priority like previous modes. If user has a sensitive system where all the three metrics i.e. resource, throughput and power are considered then this mode will help the user to select the appropriate cipher suite according to its hard metrics budget.

2.1. Efficient System Index Evaluation(ESI)

Here we are proposing a new factor named Efficient System Index(ESI) which is a single parameter to measure the efficiency of all possible crypto combination of SSL/TLS system. Lets see how it comes.

Let P , T and R are the power, throughput and resource of the crypto algorithm respectively. Now

$$\begin{aligned} ESI &\propto T, \\ ESI &\propto 1/P, \\ ESI &\propto 1/R, \end{aligned}$$

$$\text{Finally } ESI \propto T/PR \quad \text{For a given algorithm} \quad (1)$$

Considering the unites of P , T and R the said straight forward ESI might not work properly. The different units of measurement of Power(mili watt), Throughput(Mbps) and resource (no. of slices) could have significant effect on ESI by shadowing others. Normalization technique is a standard well know method to scale those 3 parameters from 0 to 1 value.

Instead of having P , T and R directly, we would take P/P_{max} , T/T_{max} and R/R_{max} where P_{max} , T_{max} and R_{max} represent maximum value of Power, Throughput and Resource respectively. We need to inverse the scaling range of P and R , as they are inversely proportional to ESI. So finally the ESI look like this

$$ESI = (1 - \frac{P}{P_{max}}) + \frac{T}{T_{max}} + (1 - \frac{R}{R_{max}}). \quad (2)$$

Now before evaluating ESI of all possible combinations of SSL/TLS we need to generate the power, throughput, and resource table from those same combinations. Let $E[3 \times n]$, $H[3 \times m]$ and $K[3 \times l]$ are 3 matrices representing performance metrics of encryption, hash and key exchange algorithms respectively. The 1st, 2nd and 3rd columns of these matrices represent power (mw), throughput (Mbps), and resources (no. of slices) respectively. 3 elements of

each row in matrices E , H and K represent power, throughput and resource of encryption, hash and key exchange algorithm respectively. e.g. The elements of 3rd row of E matrix, P_{3E} , T_{3E} & R_{3E} represent the power, throughput and resource respectively of the 3rd encryption algorithm of encryption list. Matrices E , H and K have n , m and l number of rows respectively, as we have n number of encryption algorithms, m number of hash algorithms and l number of key exchange algorithms. In this work we have chosen $n=7$, $m=3$ and $l=3$. As per the SSL and TLS cipher suite referred in [7], all encryption, hash and key exchange algorithms are disjoint, so all combinations of algorithms represented in P , T and R matrices are practically possible.

$$E = \begin{bmatrix} P_{1E} & T_{1E} & R_{1E} \\ P_{2E} & T_{2E} & R_{2E} \\ P_{3E} & T_{3E} & R_{3E} \\ \dots & \dots & \dots \\ P_{nE} & T_{nE} & R_{nE} \end{bmatrix}$$

$$H = \begin{bmatrix} P_{1H} & T_{1H} & R_{1H} \\ P_{2H} & T_{2H} & R_{2H} \\ P_{3H} & T_{3H} & R_{3H} \\ \dots & \dots & \dots \\ P_{mH} & T_{mH} & R_{mH} \end{bmatrix}$$

$$K = \begin{bmatrix} P_{1K} & T_{1K} & R_{1K} \\ P_{2K} & T_{2K} & R_{2K} \\ P_{lK} & T_{lK} & R_{lK} \end{bmatrix}$$

Inside the iterative loops in line numbers 5, 6 and 7 of the algorithm 1 matrices $P[n \times m \times l]$, $T[n \times m \times l]$ and $R[n \times m \times l]$ have been generated to store power, throughput and resource of every combination of encryption, hash and key exchange algorithms. The logical index for each cell of the three matrices generates memory references for the corresponding encryption, hash and key exchange algorithms, e.g. P_{012} refers the additive power of 0th encryption, 1st hash and 2nd key exchange algorithm respectively. Figure 1 is the pictorial view of the Power matrix. Rest of the two 3d matrices i.e. $T[n \times m \times l]$ and $R[n \times m \times l]$ will look identical as Figure 1.

According to the requirement of user the priority of power, throughput and resource should be flexible. To put more priority to any of these three factor, we introduce 3 weighted value W_p , W_t and W_r to define the priority of power, throughput and resource respectively. Higher priority has been defined as more weighted value. After imposing the priority factor equation 2 is altered as

$$ESI = W_p \times \left(1 - \frac{P}{P_{max}}\right) + W_t \times \frac{T}{T_{max}} + W_r \times \left(1 - \frac{R}{R_{max}}\right). \quad (3)$$

As the formula are normalized $W_p + W_t + W_r = 1$. Using equation 3 the Line 11 of algorithm 1 is evaluating the cut off ESI (ESI_t). It looks like

$$ESI_t = W_p \left(1 - \frac{P_{avg}}{P_{max}}\right) + W_t \frac{T_{avg}}{T_{max}} + W_r \left(1 - \frac{R_{avg}}{R_{max}}\right) \quad (4)$$

where the $P_{avg} = \frac{1}{n \times m \times l} \sum_{k=1}^{k=l} \sum_{j=1}^{j=m} \sum_{i=1}^{i=n} P_{ijk}$

$T_{avg} = \frac{1}{n \times m \times l} \sum_{k=1}^{k=l} \sum_{j=1}^{j=m} \sum_{i=1}^{i=n} T_{ijk}$

$R_{avg} = \frac{1}{n \times m \times l} \sum_{k=1}^{k=l} \sum_{j=1}^{j=m} \sum_{i=1}^{i=n} R_{ijk}$

In the 2nd set of loops starting from line no. 12, 13 and 14 ESI of each set of cipher suite combinations are being calculated by equation 3, and after that at line no 15 calculated ESI has been compared to previous ESI_t to suggest eligible cipher suite combinations. Considering the priority of power, throughput and resource (W_p , W_t and W_r) 3 category has been set. Only one decimal digit at left side of point has been considered as more than one decimal digit after point could not effect the ESI parameter significantly. The priority categories of priority classification

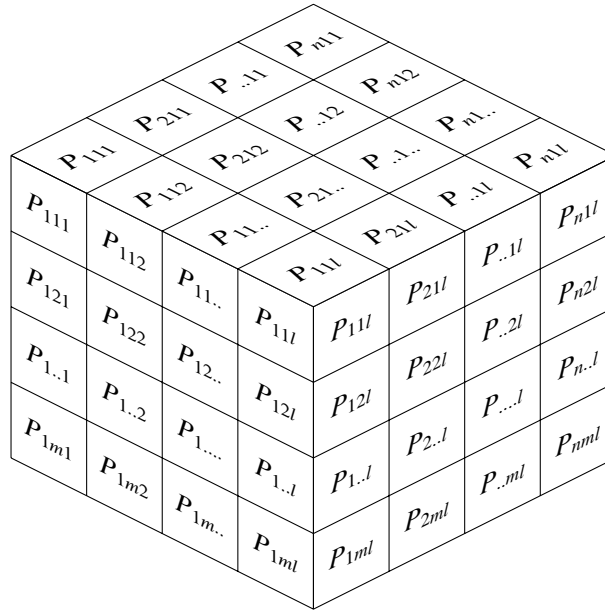


Figure 1. Power Matrix

1. *equal priority*: where W_p , W_t and W_r has same weight. $W_p=W_t=W_r$.
2. *single priority*: where system cares only about single parameter priority over the rest of the parameters. e.g. if we have only power priority then $W_p=1$, $W_t=0$ and $W_r=0$. This is relevant to section2 Power, Resource and throughput Priority mode.
3. *multiple priority*: when system is concerned about more than one parameter. For such cases all weight $W_p \neq 0$, $W_t \neq 0$ and $W_r \neq 0$. This is relevant to section2 Priority mode.

Table 1 shows the 3 categories of priority where 1st instance is for equal priority, 2nd to 4th are for single priority and 5th to 46th are for multiple priority category. For 1st instance, we have equal priority on power, throughput and resource where best and worst algorithm combinations are DES+MD5+RSA and AES+SHA256+DH_RSA respectively depending on its ESI values. For 2nd, 3rd and 4th instances the priorities are on power, throughput and resource respectively. As we have only priority on power in 2nd instance, only W_p is '1' and rest of the weight W_t and W_r are '0'. Following the same contrast W_t and W_r are 1 at 3rd and 4th rows respectively. If you see tables 2, 3 and 4 of section 5 you can justify the best and worst algorithm combination of instances 1st, 2nd, 3rd and 4th rows at table 1.

3. System Overview

The proposed hardware architecture of SSL/TLS protocol is a prototype single chip solution for parallel, multi way data communication, which may be a replacement of SSL software application in our general purpose computer and low budget embedded gadget. The proposed system would be placed between a network and embedded system/general purpose machine. Two processing elements are dedicated to receive/send data from/to special/general purpose machine and any network. The PCI and Ethernet interface is used to communicate special/general purpose machine with PE1 and network with PE2 respectively. There may be some alternative interfaces of PCI while special purpose dedicated system will be used instead of general purpose machine. There are two data flow paths.

- The data coming from PCI, will be written in the on-chip-memory using DMA, next the Crypto Engine1 (CE1) encrypts the data fetching it from the memory. The encrypted data buffer gets written on the memory again using DMA. The Processor Element 2 (PE2) reads memory through another DMA process and then sends it to the network by the ethernet interface.

Algorithm 1 Calculation of ESI

```

1: procedure InputMatrix( $E[3 \times n]$ ;  $H[3 \times m]$ ;  $K[3 \times l]$ )
2:   for  $i = 1 \rightarrow n$  do
3:     for  $j = 1 \rightarrow m$  do
4:       for  $k = 1 \rightarrow l$  do.
5:          $P[i,j,k] = E[i,0] + H[j,0] + K[k,0]$ ;
6:          $T[i,j,k] = E[i,1] + H[j,1] + K[k,1]$ ;
7:          $R[i,j,k] = E[i,2] + H[j,2] + K[k,2]$ ;
8:       end for
9:     end for
10:   end for
11:    $ESI_t = W_p(1 - \frac{P_{avg}}{P_{max}}) + W_t \frac{T_{avg}}{T_{max}} + W_r(1 - \frac{R_{avg}}{R_{max}})$ 
12:   for  $i = 1 \rightarrow n$  do
13:     for  $j = 1 \rightarrow m$  do
14:       for  $k = 1 \rightarrow l$  do
15:          $ESI[i,j,k] = W_p(1 - \frac{P[i,j,k]}{P_{max}}) + W_t \frac{T[i,j,k]}{T_{max}} + W_r(1 - \frac{R[i,j,k]}{R_{max}})$ 
16:         if  $ESI_t < ESI[i,j,k]$  then
17:           \\\Eligible Combinations are:
18:           -----
19:           Encryption_Algorithm[i]
20:           Hash_Algorithm[j]
21:           Key_Exchange_Algorithm[k]
22:           -----
23:         end if
24:       end for
25:     end for
26:   end for
27: end procedure

```

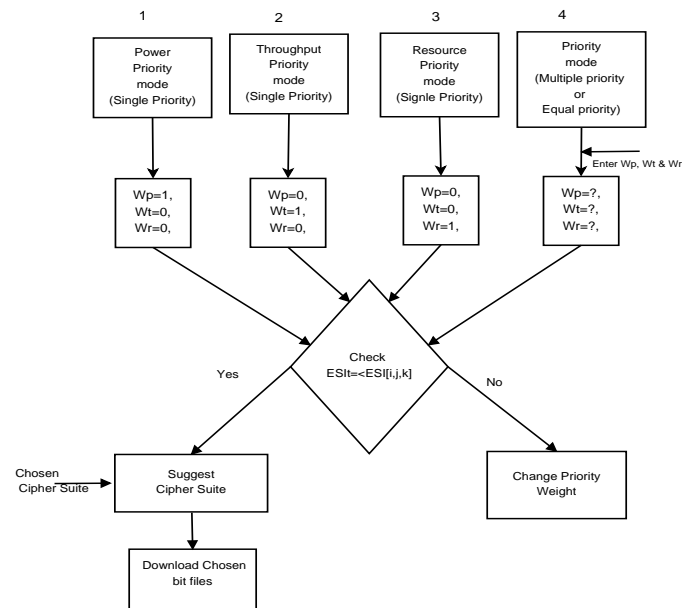


Figure 2. Modes of Preferential Algorithm

Table 1. Preferential algorithm

Sl.	Weight			Priority	ESI_i	Combination		% of Eligible Combination
	W_p	W_i	W_r			Best	Worst	
1	0.333	0.333	0.333	equal	0.3398	DES+MD5+RSA	AES+SHA256+DH_RSA	46
2	1	0	0	Single	0.3098	Idea+MD5+RSA	AES+SHA512+DH_anon	71.4
3	0	1	0	Single	0.3713	DES+SHA512+RSA	Idea+SHA256+DH_RSA	38.1
4	0	0	1	Single	0.3384	Grain+MD5+RSA	AES+SHA512+DH_RSA	66.6
5	0.8	0.1	0.1	multiple	0.3188	DES+MD5+RSA	AES+SHA512+DH_RSA	68.2
6	0.7	0.2	0.1	multiple	0.325	DES+MD5+RSA	AES+SHA512+DH_RSA	65
7	0.7	0.1	0.2	multiple	0.3217	DES+MD5+RSA	AES+SHA512+DH_RSA	68.2
8	0.7	0.15	0.15	multiple	0.3233	DES+MD5+RSA	AES+SHA512+DH_RSA	68.2
9	0.6	0.2	0.2	multiple	0.3278	DES+MD5+RSA	AES+SHA512+DH_RSA	65
10	0.6	0.3	0.1	multiple	0.3311	DES+MD5+RSA	AES+SHA256+DH_RSA	52.3
11	0.6	0.1	0.3	multiple	0.3245	DES+MD5+RSA	AES+SHA512+DH_RSA	71.4
12	0.5	0.3	0.2	multiple	0.3339	DES+MD5+RSA	AES+SHA256+DH_RSA	55.5
13	0.5	0.2	0.3	multiple	0.3306	DES+MD5+RSA	AES+SHA512+DH_RSA	65
14	0.5	0.25	0.25	multiple	0.3323	DES+MD5+RSA	AES+SHA512+DH_RSA	58.7
15	0.5	0.4	0.1	multiple	0.3373	DES+MD5+RSA	AES+SHA256+DH_RSA	42.8
16	0.5	0.1	0.4	multiple	0.3274	DES+MD5+RSA	AES+SHA512+DH_RSA	71.4
17	0.4	0.3	0.3	multiple	0.3368	DES+MD5+RSA	AES+SHA512+DH_RSA	55.5
18	0.1	0.8	0.1	multiple	0.3618	DES+SHA512+RSA	Idea+SHA256+DH_RSA	41.2
19	0.2	0.7	0.1	multiple	0.3557	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
20	0.1	0.7	0.2	multiple	0.3586	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
21	0.15	0.7	0.15	multiple	0.3571	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
22	0.2	0.6	0.2	multiple	0.3524	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
23	0.3	0.6	0.1	multiple	0.3496	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
24	0.1	0.6	0.3	multiple	0.3553	DES+SHA256+RSA	AES+SHA256+DH_RSA	42.8
25	0.3	0.5	0.2	multiple	0.3462	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
26	0.2	0.5	0.3	multiple	0.3491	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
27	0.4	0.5	0.1	multiple	0.3434	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
28	0.1	0.5	0.4	multiple	0.352	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
29	0.25	0.5	0.25	multiple	0.3477	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
30	0.3	0.4	0.3	multiple	0.3430	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8
31	0.1	0.1	0.8	multiple	0.3388	Grain+MD5+RSA	AES+SHA512+DH_RSA	69.8
32	0.2	0.1	0.7	multiple	0.3359	DES+MD5+RSA	AES+SHA512+DH_RSA	69.8
33	0.1	0.2	0.7	multiple	0.3421	DES+MD5+RSA	AES+SHA512+DH_RSA	68.2
34	0.15	0.15	0.7	multiple	0.339	DES+MD5+RSA	AES+SHA512+DH_RSA	71.4
35	0.2	0.2	0.6	multiple	0.3392	DES+MD5+RSA	AES+SHA512+DH_RSA	68.2
36	0.3	0.1	0.6	multiple	0.3331	DES+MD5+RSA	AES+SHA512+DH_RSA	69.8
37	0.1	0.3	0.6	multiple	0.3454	DES+MD5+RSA	AES+SHA512+DH_RSA	53.9
38	0.3	0.2	0.5	multiple	0.3364	DES+MD5+RSA	AES+SHA512+DH_RSA	68.2
39	0.4	0.1	0.5	multiple	0.3302	DES+MD5+RSA	AES+SHA512+DH_RSA	71.4
40	0.1	0.4	0.5	multiple	0.3487	DES+MD5+RSA	AES+SHA256+DH_RSA	42.8
41	0.2	0.3	0.5	multiple	0.3425	DES+MD5+RSA	AES+SHA512+DH_RSA	55.8
42	0.25	0.25	0.5	multiple	0.3394	DES+MD5+RSA	AES+SHA512+DH_RSA	65
43	0.3	0.3	0.4	multiple	0.3397	DES+MD5+RSA	AES+SHA512+DH_RSA	55.5
44	0.4	0.2	0.4	multiple	0.3335	DES+MD5+RSA	AES+SHA512+DH_RSA	65
45	0.2	0.4	0.4	multiple	0.3458	DES+MD5+RSA	AES+SHA256+DH_RSA	42.8
46	0.4	0.4	0.2	multiple	0.3401	DES+SHA512+RSA	AES+SHA256+DH_RSA	42.8

- The encrypted data coming from Ethernet, gets written in the on chip memory using DMA. That data buffer is fetched by the Crypto Engine2 (CE2) for the decryption process. The decrypted data gets written back to the memory using DMA. The Processing Element 1 (PE1) will read that data buffer using another DMA and sends it to the system using the PCI interface.

The architectural bird view has been shown in Figure 3. A brief overview of the different blocks of the proposed architecture is given below.

3.1. Processing Element

Two 32 bit ARM cores are used as Processing Elements (PEs) to handle the data coming from PCI, Ethernet interfaces and forwarding the data back to the memory using DMA module. PE1 selects the suitable algorithm combination using preferential algorithm to download the corresponding bit files of chosen algorithm combination into the partitioned partial region of the FPGA from the Flash memory. PE2 is responsible for key exchange procedure with server using Key Exchange Block. Both PE1 and PE2 can also synchronize the whole pipeline process using the Process Scheduler IP (PS). Microblaze processor may be also an alternative of ARM processor as PE.

3.2. Process Synchronizer (PS)

PS is a custom logic IP, which synchronizes and monitors the whole system flow and system status using its master bus and slave bus respectively. The PS is connected with AXI Streamer and Crypto Engine through controlling I/Os, which act like flag bits inside PS. PEs, AXI Streamer and Crypto Engine can read and write those flags through master bus and slave bus of PS respectively. During a full data process cycle flags are being updated by *done* signals of each process like *writeDMA* process, *readDMA* process and *crypto* process. PEs read those *done* flags using API's of PS through AXI bus and according to the status of those *done* signals PEs synchronize the *start* signals of those processes. Two PS has been used by two PEs. The PS can be used for debugging purpose also to monitor the whole system status.

3.3. Crypto Engine

Each crypto engines consists two blocks, hashing block, and encryption block. Two crypto engines have been used in two sides of the proposed architecture. One is dedicated for ciphering and hashing of plain texts coming from PCI, another is used for deciphering and hash checking of encrypted text coming from the Ethernet.

3.4. Key Exchange Blocks

Before starting the data communication between client and server key exchange is necessary. Since server data comes through the Ethernet interface, which is controlled by PE2, so key exchange block is also connected with PE2.

3.5. AXI Streamer

AXI Streamer is used as a interface controller of crypto engine and DMA. DMA uses a master and slave bus for read write operation and few controlling ports to control the bus status. The details of the communication between processors and crypto engine through DMA and AXI Streamer is shown in Figure 4.

3.6. ICAP and Flash Memory

The Internal Configuration Access Port (ICAP) is a IP provided by xilinx to read and write bit files from any storage memory. After selection of appropriate cipher suite by preferential algorithm, the bit files of the chosen key exchange, encryption and hash algorithm gets configured from flash memory. The whole process is controlled by PE1. Details shown in [8] The flow of the steps are show in Figure 6.

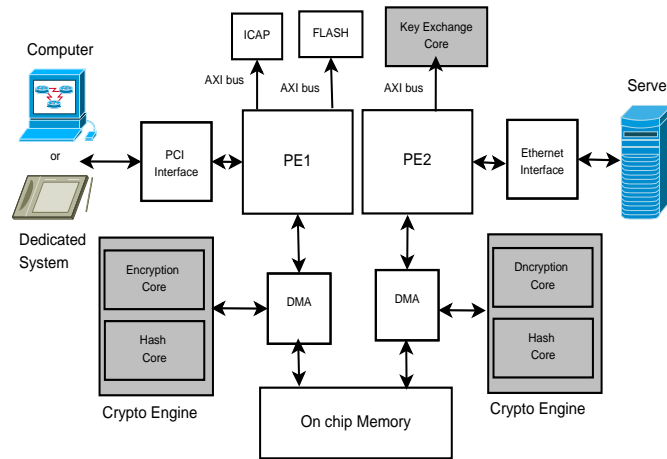


Figure 3. System Architecture

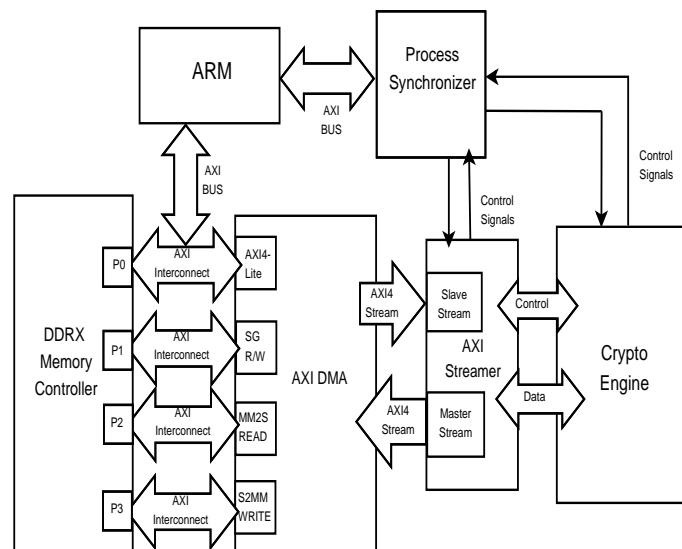


Figure 4. Processor and Crypto Engine Communication

4. Architecture Topology

Former topology shown in Figure 5(a) has been adopted by many literatures like [1], [2], [3] and [4] where a single 32 bit bidirectional bus has been used to communicate between PCI and crypto engines. Bidirectional data bus performs both data read and write operation. Even huge amount of arbitrations has been incorporated in this kind of architecture, still the data is being congested as shown in Figure 7(c). The entire bus is being busy until a full data transfer has been completed. In [5] a pipelined architecture has been proposed where two different 64 bit bus is used for reading and writing operation, as a result both operations can be done at the same time. The authors used a single PCI interface which accepts and forwards the data after processing through the same PCI channel, which results to congestion. In Figure 7(b) for 5th packet time t has been compromised due the congestion of data in the PCI. Generally in most of the application systems (client) Ethernet port is the interface for transmission and reception of data. Hence, we have used of two separate interfaces as shown in Figure 7(a). As two separate interfaces has been used, collision of data receiving process and transmitting process has been solved in our proposed architecture, which causes an acceleration in cryptographic processes over receiving and transmitting way data path.

4.1. PCI to Ethernet

Receive plain text from system through PCI for encryption and hashing process and send it to Ethernet. From stage1 to 3 of figure 6 data are coming from PCI to Crypto engine through a Write DMA(WDMA). At stage 4, Read DMA(RDMA) read data from said crypto engine and at stage 5 send it to ethernet Interface.

4.2. Ethernet to PCI

Receive cipher text from ethernet for decryption and hash checking process and send it to system through PCI. Rest of the architecture topology is symmetric with previous data flow. As two separate interfaces has been used, collision of data receiving process and transmitting process has been solved in our proposed architecture

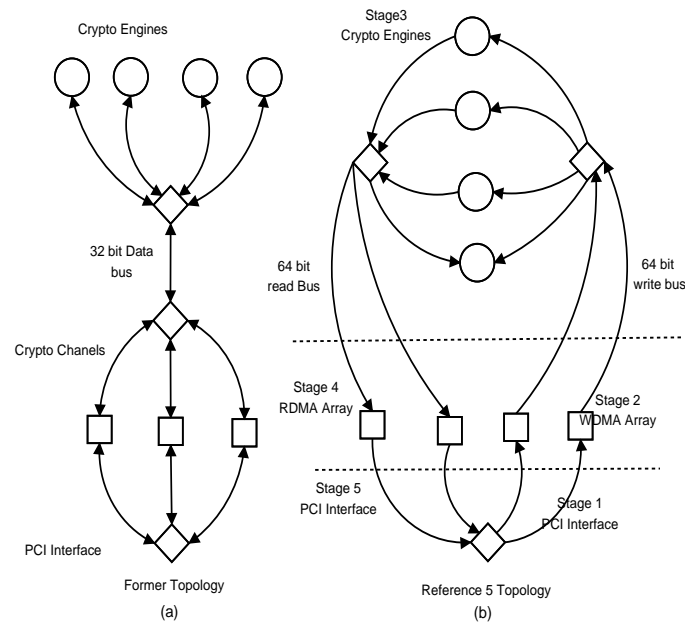


Figure 5. Existing Topology

5. Result and Implementation

The proposed SSL/TLS cipher suite has been implemented with 3 hashing algorithms, 7 encryption algorithms and 3 key exchange algorithms. Different design metrics of these algorithms are shown in table 2, 3 and 4. The possible combinations for the 3 types of algorithms are $7 \times 3 \times 3 = 63$.

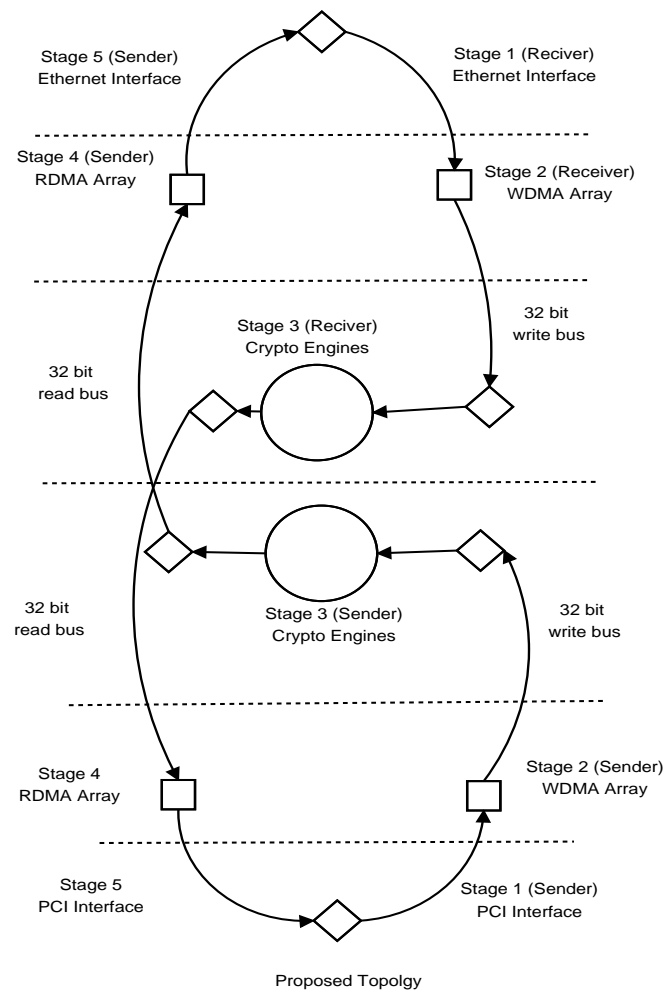


Figure 6. Proposed Topology

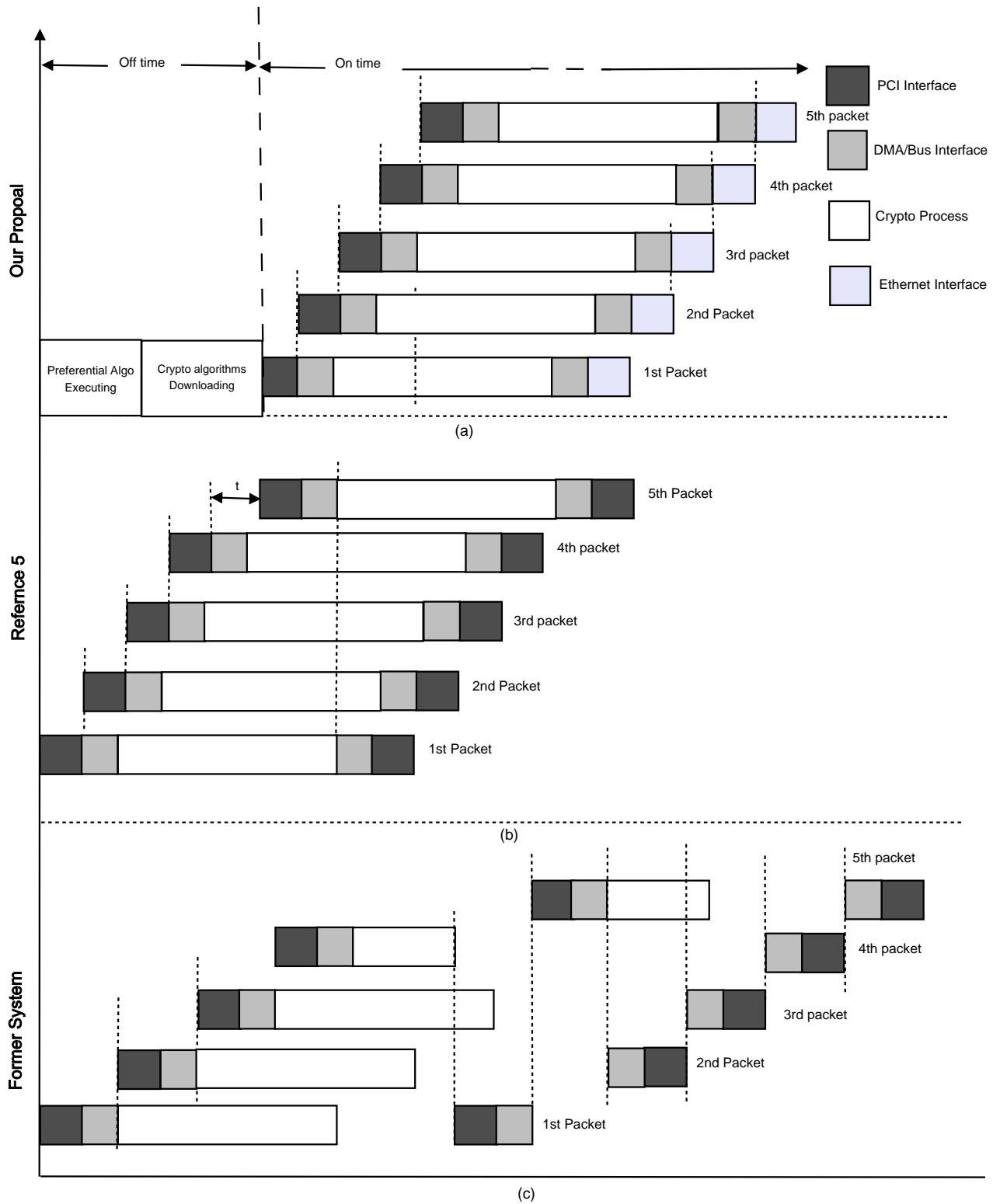


Figure 7. Data transfer efficiency comparison.

Table 2. Onchip resource throughput power of HASH algorithm

Name of hash	Slice #	Power (mw)	Throughput Gbps	Critical path(ns)
SHA-256	1385	176	0.735	3.85
SHA-512	2647	278	1.471	5.50
MD5	992	112	0.916	7.31

Table 3. Onchip resource throughput power of Crypto algorithm

Name of Encryption	Slice #	Power (mw)	Throughput Gbps	Critical path(ns)
AES	11385	1183	1.067	2.939
RC4	5383	994	0.931	4.127
Grain	237	99.7	0.116	1.689
Salsa	2839	107	3.725	2.064
DES	456	103	7.45	2.468
3DES	1478	117	2.48	2.468
Idea	320	95	0.079	8.18

Table 4. Onchip resource throughput power of Key Exchange algorithm

Name of hash	Slice #	Power (mw)	Throughput Gbps	Critical path(ns)
RSA	13910	1589	0.298	3.85
DH_anon	14012	1767	0.149	3.89
DH_RSA	14789	1918	0.099	5.67

Table 5. Comparison Table

Name	[5]	[2]	[1]	[9]	[10]	This work
Device	xc3s500e	ASIC	ASIC	xc2vp100	xcv1000e	xc7z020
Clock MHz	150	66	80	106.6	24.2	125
AES Gbps	2.042 /0.951	–	1.462	1.2	0.310	1.067
DES Gbps	2.453	1.13	–	–	–	7.45
3DES Gbps	0.663	–	–	–	–	2.48
RSA	2350/s	520/s	26/s	–	–	0.0391gb/s

6. Conclusion

In this paper practical application focusing the adoption of high speed pipelined NSP architecture to accelerate the performance of SSL/TLS protocol is proposed, which can be used in general purpose and also in special purpose machines. The partitioned partial region where algorithm combinations are being dumped must be made of FPGA, but rest of the system can be made of ASIC or of FPGA. The partial reconfiguration feature of FPGA can dynamically alter the security paradigms depending upon the choice of the proposed preferential algorithm regarding the requirement of power, throughput, resource and ESI of the system. The results obtained following the implementation of the proposed architecture show that it has better pipelined process, better hardware flexibility and well optimized resource-power parameters in comparison to what was achieved in any of the existing literatures.

References

- [1] C.-H. Wang, C.-Y. Lo, M.-S. Lee, J.-C. Yeh, C.-T. Huang, C.-W. Wu, S.-Y. Huang, A network security processor design based on an integrated soc design and test platform, in: Design Automation Conference, 2006 43rd ACM/IEEE, 2006, pp. 490–495. doi:10.1109/DAC.2006.229266.
- [2] Motorola, Mpc 190 security processor fact sheet motorola 2003, publishers-freescale semiconductor.
URL http://www.freescale.com/files/netcomm/doc/fact_sheet/MPC190FACT.pdf
- [3] BROADCOM, Broadcom (2004). bcm 5840 gigabit security processor.
URL <http://www.broadcom.com/collateral/pb/5840-PB03-R.pdf>
- [4] BROADCOM, Hifn 2008.hifn 7954 security accelerator data sheet.
URL http://www.hifn.com/uploadedFiles/Library/Product_Briefs/7954.pdf
- [5] H. Wang, G. Bai, H. Chen, A gbps ipsec ssl security processor design and implementation in an fpga prototyping platform, Journal of Signal Processing Systems 58 (3) (2010) 311–324. doi:10.1007/s11265-009-0371-2.
URL <http://dx.doi.org/10.1007/s11265-009-0371-2>
- [6] Xilinx, Field programmable gate array (fpga).
URL <http://www.xilinx.com/fpga/>
- [7] B. A. Forouzan, Cryptography and Network Security, second reprint 2011 Edition, Tata McGraw-Hil, ISBN-13: 978-0-07-070208-01, New Delhi, India, 2011.
- [8] Xilinx, Axi hardware icap.
URL http://www.xilinx.com/products/intellectual-property/axi_hwicap.htm
- [9] J. Lu, J. Lockwood, Ipsec implementation on xilinx virtex-ii pro fpga and its application, in: Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, 2005, pp. 158b–158b. doi:10.1109/IPDPS.2005.262.
- [10] M. McLoone, J. McCanny, A single-chip ipsec cryptographic processor, in: Signal Processing Systems, 2002. (SIPS '02). IEEE Workshop on, 2002, pp. 133–138. doi:10.1109/SIPS.2002.1049698.